

Improving package recommendations through query relaxation

Matteo Brucato*

Azza Abouzied§

Alexandra Meliou*

*School of Computer Science
University of Massachusetts
Amherst, USA

{matteo,ameli}@cs.umass.edu

§Computer Science
New York University
Abu Dhabi, UAE
azza@nyu.edu

ABSTRACT

Recommendation systems aim to identify items that are likely to be of interest to users. In many cases, users are interested in *package recommendations* as collections of items. For example, a dietitian may wish to derive a dietary plan as a collection of recipes that is nutritionally balanced, and a travel agent may want to produce a vacation package as a coordinated collection of travel and hotel reservations. Recent work has explored extending recommendation systems to support packages of items. These systems need to solve complex combinatorial problems, enforcing various properties and constraints defined on sets of items. Introducing constraints on packages makes recommendation queries harder to evaluate, but also harder to express: Queries that are under-specified produce too many answers, whereas queries that are over-specified frequently miss interesting solutions.

In this paper, we study query relaxation techniques that target package recommendation systems. Our work offers three key insights: First, even when the original query result is not empty, relaxing constraints can produce preferable solutions. Second, a solution due to relaxation can only be preferred if it improves some property specified by the query. Third, relaxation should not treat all constraints as equals: some constraints are more important to the users than others. Our contributions are threefold: (a) we define the problem of deriving package recommendations through query relaxation, (b) we design and experimentally evaluate heuristics that relax query constraints to derive interesting packages, and (c) we present a crowd study that evaluates the sensitivity of real users to different kinds of constraints and demonstrates that query relaxation is a powerful tool in diversifying package recommendations.

Categories and Subject Descriptors

H.2.4 [Information Systems]: Systems—*Query processing*

Keywords

recommendation system; packages; query relaxation

1. INTRODUCTION

The rapid growth of available data has created unique challenges for the field of database research, and has forced us to rethink multiple aspects of data management. Most existing work that deals with *volume* in Big Data targets two directions: increasing data size (more rows) and increasing data dimensionality (more columns). Along these two directions, database research has explored time-efficient algorithms that handle expanding data sizes, and machine learning research has focused on the inference challenges introduced by the

growth in richness and dimensionality of the data. In this paper, we study a third direction in Big Data that has so far received little attention: assembling data items into different possible collections of items, or *packages* (more combinations) [3, 5]. From a set of n items, one could identify up to 2^n subsets with some desired property; this makes construction of packages extremely challenging, even for small datasets. Effectively, even a dataset with only 100 items can introduce Big Data challenges for a package recommendation system.

This complexity has deterred data systems from providing full-fledged support for deriving packages. Yet, the need for package support arises in many applications. For example, most airline travel today includes non-direct itineraries that need to combine several legs, which, as a package, comply with the budget, schedule, and airline preferences of the traveler. Package recommendation systems have been used to derive travel plans [13], team formations [7], course combinations [9, 10], and nutritionally balanced meal plans [3]. The items in these packages need to satisfy criteria for the individual items in the collection, as well as constraints defined on the entire set of items in the package.

Introducing constraints on collections of items makes recommendation queries harder to evaluate: In contrast with traditional query evaluation, where each individual item is checked against the query conditions, when evaluating package queries it is not feasible to examine all possible packages, as the number of combinations grows exponentially [3, 9, 12, 14].

Moreover, this complexity poses significant *usability* challenges. Package queries that are under-specified produce too many answers, and queries that are over-specified frequently miss interesting solutions. Consequently, package recommendation systems often produce unsatisfactory results: Would a traveler still prefer her usual carrier if a different airline offered a travel plan with fewer stops and at a much lower price? In cases like this, the user often prefers to relax the airline constraint to improve travel time and airfare.

In this paper, we propose a novel approach to producing package recommendations that is based on *query relaxation*. In contrast with existing recommendation systems that search for the top- k packages which satisfy a given set of constraints, we show that relaxing the package constraints can often produce preferable solutions. For example, removing the airline constraint from a travel plan recommendation query may result in a shorter travel time. We support our approach with a crowd user study. Our study showed that dissatisfaction with the package recommendations that strictly adhere to user's preferences was very common (30% of cases). This is evidence that query relaxation is necessary to provide satisfactory package recommendations. More stunningly, our relaxation results had overall higher approval rates (76%) than the non-relaxed packages (71%). Our study also showed that users had different

sensitivity to relaxing different types of constraints. This indicates that relaxation algorithms can be more effective if they prioritize constraints based on this sensitivity level.

Our treatment of query relaxation in this context is unique: existing systems use relaxation to address empty result sets, rather than to derive alternative recommendations. In summary, our work offers three key insights: First, relaxing constraints can produce preferable solutions even when the original query result is not empty. Second, a solution due to relaxation can only be preferred if it improves some property specified by the query. Third, relaxation should not treat all constraints as equals: some constraints are more important to the users than others. We organize our contributions as follows:

Section 2. We define the problem of deriving package recommendations through query relaxation. This is a novel use of query relaxation, compared to existing work.

Section 3. We study the effect of coarse-grained relaxations (removal of constraints) to package results. We discuss relaxation approaches that aim to improve a package’s utility, and approaches that balance the improvement in utility with the error due to relaxation. We further present greedy heuristics that perform comparably to the best-case approaches, and we show that relaxation can be effective even when it involves a very small number of constraints.

Section 4. We present a crowd study that verifies our intuition that query relaxation can produce better recommendations and demonstrates that users accept relaxations of certain kinds of constraints more frequently than others.

Section 5. We discuss related work in recommendation systems and query relaxation. We note that our approach is a novel way of improving package recommendation results and differs from the traditional use of query relaxation.

Section 6. Our approach showed great promise in our user study. Our goal is to refine our query relaxation methods to make relaxations more targeted and fine-grained; we discuss several directions that we plan to pursue towards this goal.

2. RECOMMENDATIONS VIA RELAXATION

In this section, we define the problem of improving package recommendations using query relaxation. A package recommendation query typically involves three types of constraints:

1. *Base constraints* describe restrictions on individual items in the package (e.g., each meal should be gluten free, each flight leg should be no longer than 4 hours, etc). Base constraints are regular selection predicates in SQL.
2. *Global constraints* describe restrictions on the package as a whole (e.g., the meal plan should have no more than 1,500 calories in total, the entire trip should cost less than \$2,000, etc). These correspond to global constraints in CourseRank [9] and PackageBuilder [3], and to compatibility constraints in [5].
3. *Cardinality constraints* are a special type of global constraints that restrict the desired number of items in a package.

In this paper, we focus on *conjunctive* package recommendation queries, i.e., queries that do not contain disjunction or negation of constraints. Some of the relaxation results that we discuss in this paper generalize to other cases as well, but we leave a thorough study of general query relaxations for future work.

In addition to constraints, package recommendation queries also include an *objective function*, which is sometimes called *objective criterion*, *objective clause* [3], *score* [9], or *utility* [5]. The objective function describes an attribute that the package recommendation should either minimize or maximize (e.g., minimizing the total airfare, maximizing the amount of protein in a diet, etc).

EXAMPLE 1. A dietitian would like to use a meal planner application to put together a nutritious and balanced collection of meals for a client. She would like to restrict each meal to no more than 60mg of cholesterol (base constraint); she would like each daily plan to include 3 or 4 different meals (cardinality constraint) and at least 1,500 calories in total (global constraint). To trust that the client will follow the dietary plan, she would like to minimize the total preparation time needed (objective function).

An answer to the query of Example 1 is a collection of meals that satisfies all base and global constraints. Traditional recommendation systems typically return the *top-k* recommendations, which means that out of all the packages that satisfy the constraints, they would return the *k* packages with the least preparation time. We challenge this rigid view on recommendations: If we provide the dietitian with a package where one meal contains 65mg of cholesterol, but with a drastically lower preparation time, she might prefer this package to the top-1 non-relaxed result.

We propose to provide *diverse package recommendations* by relaxing the package constraints. For this paper, we focus on *coarse* relaxations that remove constraints entirely, rather than changing their values.¹ In defining the optimal relaxation, we apply two intuitions: (a) the optimal relaxation should improve the value of the objective function compared to the top-1 non-relaxed result, and (b) it should minimize the total deviation from the query constraints.

Optimal package relaxation. We denote a package recommendation query as $Q_{C,\mathcal{F}}$, where $C = \{c_1, \dots, c_n\}$ is a set of base and global constraints, and \mathcal{F} is an objective function. Each constraint c_i is a predicate of the form: $f_{c_i} \text{ op } \beta_i$, where f_{c_i} is a function over all the items in a package, *op* is a comparison operator (e.g., $<$, \geq), and β_i is a constant in \mathbb{R} . For example, the global constraint in Example 1 is $\text{sum}(\text{calories}) \geq 1,500$. We denote with $f_{c_i}(Q)$ the value of f_{c_i} over the top-1 package for $Q_{C,\mathcal{F}}$. Similarly, we denote with $\mathcal{F}(Q)$ the value of the objective function over the top-1 package for $Q_{C,\mathcal{F}}$.

DEFINITION 1 (PACKAGE RELAXATION). A package query $Q_{C',\mathcal{F}}$ is a relaxation of $Q_{C,\mathcal{F}}$ if $C' \subset C$.

The top-1 result of a package relaxation may violate one or more constraints in $C \setminus C'$. We use $Q' \not\models c_i$ to denote that $Q_{C',\mathcal{F}}$ violates constraint c_i . However, as the following proposition shows, relaxations may improve the value of the objective function:

PROPOSITION 1. Let $Q_{C',\mathcal{F}}$ be a relaxation of $Q_{C,\mathcal{F}}$. Then:

- If \mathcal{F} is a maximization objective, then $\mathcal{F}(Q') \geq \mathcal{F}(Q)$.
- If \mathcal{F} is a minimization objective, then $\mathcal{F}(Q') \leq \mathcal{F}(Q)$.

We define the *optimal relaxation* of a package query as the relaxation $Q_{C^*,\mathcal{F}}^*$ that maximizes the *improvement* in the objective function, while minimizing the deviation from the constraints of the original query. More formally:

DEFINITION 2 (OPTIMAL RELAXATION). The optimal relaxation for a package query $Q_{C,\mathcal{F}}$ is a query $Q_{C^*,\mathcal{F}}^*$ such that:

$$Q_{C^*,\mathcal{F}}^* = \arg \max_{Q_{C',\mathcal{F}}, C' \subset C} \frac{1 + I(Q'; Q)}{1 + \mathcal{E}(Q', C \setminus C')} \quad (1)$$

¹Several aspects of our work easily generalize to finer relaxations, but our results show that even coarse relaxations perform well.

The functions I and \mathcal{E} are distance metrics that measure the change in the value of the objective function and the amount of constraint violation of the relaxed query, respectively. In our implementation and experiments, we modeled I as the *percentage improvement* in the objective function: $I(Q'; Q) = \frac{|\mathcal{F}(Q) - \mathcal{F}(Q')|}{\mathcal{F}(Q)}$. We modeled \mathcal{E} as the *mean absolute percentage error* (MAPE): $\mathcal{E}(Q', C \setminus C') = \frac{1}{|C|} \sum_{c_i \in C, Q' \not\models c_i} \frac{|\beta_i - f_{c_i}(Q')|}{\beta_i}$.

3. DERIVING RELAXATIONS

Deriving the optimal relaxation Q^* (Definition 2) is computationally expensive: The space of possible relaxations is exponential in the number of constraints of Q . Moreover, computing the objective value $\mathcal{F}(Q')$ for each relaxation Q' is also an expensive operation, as it involves deriving the top-1 package for Q' . In this section, we propose simple but effective heuristics to avoid searching for Q^* in the entire space of candidate relaxations. These methods are able to find good approximations for Q^* very efficiently.

3.1 Extent of relaxation

We first study how aggressively we should relax queries: Is it better to relax more or fewer constraints? We investigate the effectiveness of relaxations when we exhaustively relax queries by removing a fixed amount of constraints. Given a query $Q_{C, \mathcal{F}}$ and a relaxation level k , we derive a relaxed query $Q_{C^k, \mathcal{F}}$, such that $|C^k|/|C| = (100 - k)\%$. We use two methods to derive Q^k based on Definition 2:

EXHAUSTIVE-I: maximizes the *improvement* $I(Q^k; Q)$, while ignoring constraint violations (i.e., $\mathcal{E}(Q^k, C \setminus C^k) = 0$).

EXHAUSTIVE-IE: maximizes the *improvement/error ratio*, similarly to Equation (1).

We tested the performance of these exhaustive relaxation approaches on a dataset of real recipes extracted from allrecipes.com. We constructed a set of 10 random package queries, in the spirit of Example 1, where the total number of constraints ranged from 3 to 10. Half of the queries included a minimization objective and half included a maximization objective.

Figure 1a shows the percentage improvement that EXHAUSTIVE-I achieves in the objective function. We see that the percentage improvement increases rapidly for small amounts of relaxation, but the improvement slows down when we relax more aggressively. The error line represents the mean absolute percentage error for all the constraints that the relaxed query violates. Following Example 1, if the top-1 solution of a relaxed query Q^k has a total of 1,000 calories, the error due to that constraint would be $\frac{|1500 - 1000|}{1500} = 33\%$.

EXHAUSTIVE-IE behaves similarly, achieving good improvements in the objective quickly, but the error is generally lower, especially for small relaxation percentages (Figure 1b). For comparison purposes, Figure 1c shows the performance of a relaxation algorithm that randomly chooses which constraints to relax.

This experiment demonstrates three important points:

1. The improvement curves rise sharply for low levels of relaxation, while the gains are reduced when we relax more aggressively. This indicates that it is sufficient to relax few of the constraints.
2. Removing constraints at random is worse than the best-case approaches for low relaxation levels. This means that relaxation cannot be arbitrary.
3. Optimizing for the improvement/error ratio can greatly reduce the error in low relaxation levels, while achieving similar values of improvement as relaxations that ignore the error.

3.2 Improving the running time

The experiment of Figure 1 shows that relaxations can achieve good improvement and error rates with the removal of a small number of constraints. This means that focusing the relaxation search on removing one or two constraints can be an effective and practical heuristic. In this section, we explore two iterative heuristics that greedily choose one constraint to remove at a time. At each iteration i , given $Q_{C^i, \mathcal{F}}$ (initially set to $Q_{C, \mathcal{F}}$ at iteration 0), each method removes:

GREEDY-I: the constraint that results in the *highest improvement*:

The algorithm selects $c_j \in C^i$ that maximizes $I(Q^{i+1}; Q^i)$, where $Q_{C^i \setminus \{c_j\}, \mathcal{F}}^{i+1}$ is obtained from Q^i by removing constraint c_j .

GREEDY-IE: the constraint that results in the *highest improvement/error ratio*: The algorithm selects $c_j \in C^i$ that maximizes

$$\frac{1 + I(Q^{i+1}; Q^i)}{1 + \mathcal{E}(Q^{i+1}, C^i \setminus C^{i+1})}.$$

Effectiveness. We evaluated the greedy heuristics on the same dataset and queries as the previous experiment. The results are presented in Figure 2. The two heuristics achieve similar improvement and error as their exhaustive counterparts, especially for low levels of relaxation (Figures 2a and 2b).

Efficiency. As expected, the greedy heuristics perform much better in terms of running time (Figure 2c). There is no runtime difference between GREEDY-I and GREEDY-IE, thus their performance is indicated with a single line “Greedy” in Figure 2c. We improve the efficiency of the greedy heuristics even further by introducing *bidirectional greedy* as an optimization: Bidirectional greedy starts from the non-relaxed query and removes one constraint at a time when the target relaxation level is below 50%. When the target relaxation level is above 50%, the heuristic starts from the empty constraint set and adds one constraint at a time, until it reaches the desired level. This optimization achieves further runtime improvements, keeping the runtime of bidirectional greedy always below exhaustive search.

4. USER-AWARE RELAXATION

In this section, we evaluate the usability of our approach with a crowdsourced user-study. In the study, we ask crowd workers to evaluate the quality of different meal-plan package recommendations with respect to a set of given nutritional requirements and other specifications. With this study, we aim to answer two research questions:

RQ1: Are users willing to accept relaxed recommendations?

RQ2: Do users have preferences with respect to the types of constraints to be removed?

We start by describing the dataset we used, the crowd task design, and the characteristics of the data we collected. We then proceed to discuss our results across these two fronts.

4.1 Experiment setup

We evaluate the quality of package recommendations for a meal planning application.

Dataset. We built a meal planner recommendation system with real recipe data collected from allrecipes.com. We collected 7,955 recipes with information on ingredients, nutritional content, and preparation time. Our application constructs meal plan recommendations given constraints on preparation time and recipe contents.

Task design. We designed a crowdsourcing task to evaluate (a) whether users like recommendations that relax one of the package

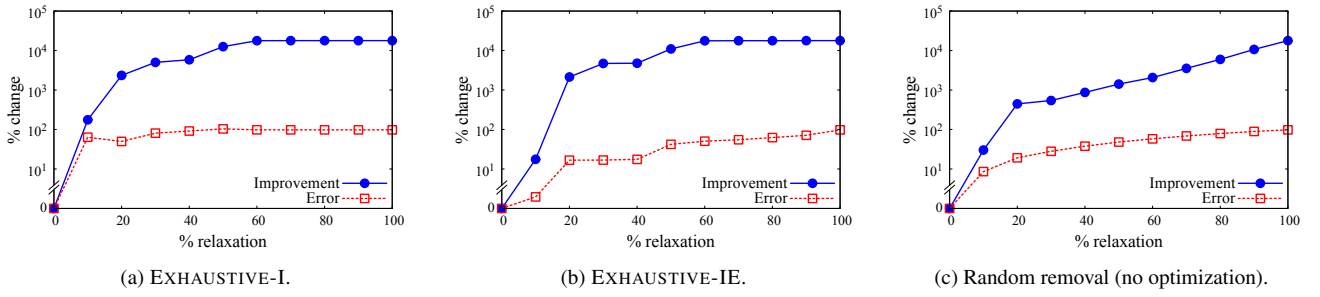


Figure 1: Improvements and errors of exhaustive and random search. Removing more constraints increases both the improvement and the error. Significant improvement can be achieved after only a few constraints are removed. Optimizing for improvement/error ratio helps reduce the amount of error when less constraints are removed.

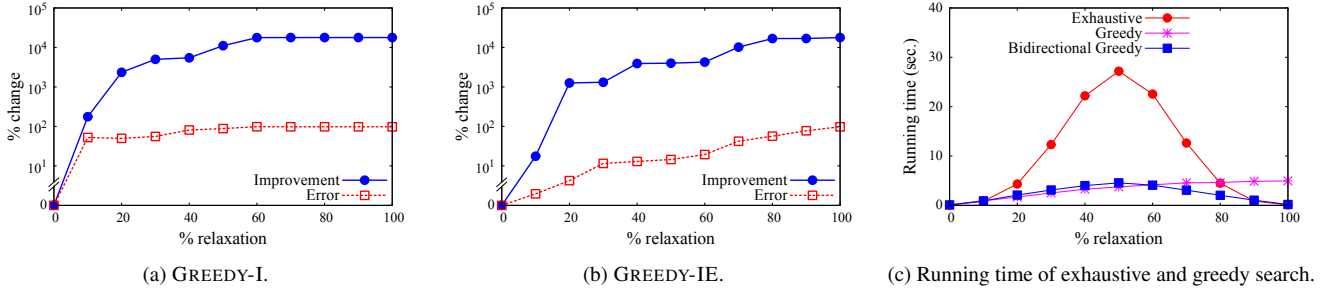


Figure 2: Performance of the greedy heuristics. Both versions of greedy search can achieve high improvements and low errors similarly to exhaustive search, while significantly reducing the running time.

constraints, and (b) whether they are sensitive to some constraints more than others. In the task scenario, we tell the crowd workers that we need to recommend some meal plans to a user, given a set of preferences that the user specified. The user specifications include four types of preferences:

Cardinality constraint: The number of meals in the plan.

Objective criterion: All task instances express that the user prefers the preparation time for the meal plan to be as low as possible.

Base constraints: Two constraints on the nutritional content of each meal in the plan.

Global constraints: Two constraints on the nutritional content of the entire meal plan.

After listing the user preferences, the task displays five alternative meal plans and asks workers to select, for each plan, whether they would recommend it to the user. Each task includes the following five meal plan types:

ORIGINAL: It strictly adheres to the constraints specified in the user preferences.

CARDINALITYRELAX: It relaxes the cardinality constraint.

BASERELAX: It relaxes one of the base constraints.

GLOBALRELAX: It relaxes one of the global constraints.

RANDOM: It is a collection of randomly selected meals that satisfies the cardinality constraint.

The plans are computed automatically, using the PACKAGE-BUILDER engine [3], and are listed in increasing order of preparation time. We use colors to indicate adherence to or violation of the constraints, which helps the workers easily note which constraints were violated. Figure 3 shows a partial screenshot of the task. We chose to omit recipe names to avoid worker bias due to personal preferences (e.g., biases due to religious dietary restrictions or cuisine).

Collected data. We automatically generated 50 different unique configurations of our task on the Crowdfunder platform.² Each configuration was completed by 10 unique workers, and each worker was not allowed to complete more than 5 configurations.

We used the explanation field to identify and remove obvious spammers from the dataset. We rejected workers who gave the same answer and comment in every task they completed, workers who entered random data in the explanation field, and workers who gave explanations that were inconsistent with their selections. After cleaning, our dataset included 115 unique workers, and 306 unique task instances.

Figure 4 summarizes our results. The first table shows the number of times that each type of plan was recommended. We note that in 16 cases, users recommended the RANDOM plan, which could mean that the dataset may still contain a few spammers that we were not able to identify. The other two tables show the number of times that each relaxation result was recommended in the cases that the ORIGINAL plan was accepted or rejected, respectively.

4.2 RQ1: Evaluation of relaxations

We first analyze our dataset to evaluate whether relaxations are useful. Our data shows that users are often dissatisfied with the meal plan that follows all of the specified constraints. From the 306 completed tasks in our dataset, the ORIGINAL plan is rejected about 30% of the time. The reason is usually that users consider the value of the objective criterion unacceptable. This means that relaxations are needed, even when the original recommendation set is not empty.

We further observe that users are in fact *more likely to choose a relaxed plan*, than the plan that strictly adheres to the specified constraints. Out of the 306 tasks, users selected at least one of

²<http://www.crowdfunder.com/>

Our user listed the following preferences, in no particular order:

- I prefer 4 meals.
- I prefer the **preparation time** to be as **low** as possible!
- I prefer that each meal has:
 - Less than 60.0 mg of cholesterol.
 - More than 2.5 g of fiber.
- I prefer that overall the plan has:
 - Less than 10.0 g of fat **in total**.
 - More than 1500.0 kcal of calories **in total**.

For each of the following meal plans, please indicate whether you would recommend it to the user with the preferences listed above.

The meal plans are **ordered by total preparation time** (with the lowest preparation times on top); the **preparation times are highlighted in blue**. Values highlighted in **red** indicate deviations from the user's declared preferences. Values highlighted in **green** indicate accordance to the user's declared preferences.

Meal Plan 1

Meals	Cholest.	Fiber	Fat	Cals	Prep. time
Meal 1	43.0 mg	5.7 g	8.5 g	541.0 kcal	15 min
Meal 2	7.0 mg	6.8 g	4.7 g	235.0 kcal	8 min
Meal 3	48.0 mg	3.4 g	31.9 g	643.0 kcal	15 min
Meal 4	3.0 mg	3.1 g	6.8 g	88.0 kcal	5 min

Totals

4 meals	101.0 mg	19.0 g	51.9 g > 10.0	1507.0 kcal	43 min
---------	----------	--------	---------------	-------------	--------

Would you recommend Meal Plan 1?

- ☐ Yes
☐ No

Briefly explain your choice about Meal Plan 1

Figure 3: In our crowd task, we provide workers with a list of user preferences and we ask them to choose whether to recommend each one of five meal plans. The five meal plans listed in the task always include one plan that satisfies the user constraints, three plans that relax one constraint each, and one random plan.

Overall			When ORIGINAL is recommended			When ORIGINAL is rejected		
method	recommended		method	recommended		method	recommended	
ORIGINAL	216	(70.59%)	any	150	(69.44%)	any	82	(91.11%)
BASERELAX	182	(59.48%)	BASERELAX	118	(54.63%)	BASERELAX	64	(71.11%)
GLOBALRELAX	94	(30.72%)	GLOBALRELAX	63	(29.17%)	GLOBALRELAX	31	(34.44%)
CARDINALITYRELAX	76	(24.84%)	CARDINALITYRELAX	47	(21.76%)	CARDINALITYRELAX	29	(32.22%)
RANDOM	16	(5.23%)						

Figure 4: Our study showed that relaxations may often be preferred to the non-relaxed solutions. Even when the ORIGINAL plan is chosen, users still recommend at least one relaxation almost 70% of the time. When ORIGINAL is not chosen, users find an acceptable result among the relaxed plans more than 90% of the time.

the relaxed plans (BASERELAX, GLOBALRELAX, or CARDINALITYRELAX) 75.82% of the time, which is more than the number of times they recommended the ORIGINAL plan.

Finally, our study shows that even when the ORIGINAL plan is selected, the users also recommend one or more of the relaxations in almost 70% of the cases. More importantly, in the many cases where the ORIGINAL plan is rejected, the users find an acceptable plan within the relaxations in more than 90% of the cases.

Conclusion: Query relaxation is a powerful technique for package recommendations, and often produces packages that are preferable to the non-relaxed solutions.

4.3 RQ2: Constraint sensitivity

When considering query relaxations, it is important to know whether a relaxation algorithm should prioritize certain constraints. In our experiment, our 115 unique users showed a clear preference

for relaxations of base constraints (BASERELAX was chosen in almost 60% of cases), and they were least likely to prefer relaxations of cardinality constraints (about 25% of cases). We observe similar behaviors when the ORIGINAL plan is either accepted or rejected.

We do not believe, however, that this finding generalizes: Depending on the application and dataset, we may observe different behaviors. For example, a user with celiac disease is unlikely to relax a base constrain on gluten.

Conclusion: The important takeaway of these observations is that, for a given application and dataset, users can strongly prefer one type of relaxation over another, and relaxation techniques should exploit this. These preferences may not always be known a priori, but they may be derived from past use-data of the system.

4.4 Additional lessons and discussion

We manually examined the explanations entered by workers in

order to understand the reasoning for selecting or rejecting a particular plan. The workers often cited lower preparation time as the reason for selecting a relaxation. For example, one user accepted a package because it was a “*close match for fiber as required and less time.*” Another user notes: “*Even thought [sic] the protein is low this is the best with a low prep time.*” This verifies our intuition that a good recommendation should improve some aspect of the original query (e.g., the objective criterion).

Our users often incorporated application-specific logic in their selections. For example, one person selected a BASERELAX plan because “*adding some proteins to the meal could be easy.*” This shows that relaxation algorithms can use application-specific knowledge to determine constraint priorities.

Finally, some user comments provided a good explanation of the strong bias that we observed toward relaxations of base constraints:

“*Since your preference is 60 mg of cholesterol per meal the overall will be 240 mg, so its okay.*”³

“*This meal plan meets most preferences. Two of the meals are lower in protein but two are high in protein which balances it out.*”

In these cases, workers applied a type of relaxation that we did not provide to them: They transformed a base constraint into a global constraint. This relaxation can be applied to base constraints on numeric values. We will explore this type of relaxation in future work.

5. RELATED WORK

Research in relational query modification (relaxation or refinement) attempts to mitigate two of the problems of the precise database answer model: (i) the empty-answer problem, or (ii) the too-many-answers problem [4, 6, 11]. Stefanidis et al. survey several techniques for encoding user preferences as soft constraints to improve the query results [11].

In contrast to its traditional use, we use query relaxation to *improve* the results of queries that do produce enough results, but whose “quality” is limited by over- or poorly- specified constraints. In such cases, queries can be slightly relaxed to allow the exploration of a larger pool of feasible solutions and, hopefully, the discovery of more interesting solutions.

Package recommendation systems have only recently received attention from the research community. They have been used to derive *travel plans* [13], *team formations* [1, 7], *course combinations* [9, 10], *composite items* [2], as well as nutritionally balanced *meal plans* [3]. Package recommendations are also close to *preference queries over sets* [14] and skyline groups [8], which additionally involve multiple objective criteria. Recent work by Deng et al. discusses the complexity of computing package recommendations and the complexity of searching for query relaxations [5].

6. CONTRIBUTIONS AND NEXT STEPS

In this paper, we proposed a novel method for deriving package recommendations based on query relaxation. Our crowd user study showed that users often prefer relaxed solutions to non-relaxed results, which indicates that this is a very promising direction for handling package recommendations.

Future research will extend our work in the following directions:

- Our current user study showed a clear bias toward relaxing base constraints. We plan to conduct a larger study to investigate what dictates user sensitivity toward different kinds of constraints.

- In this work, we focused on coarse relaxation (removing constraints) to test the viability of query relaxation as a package recommendation method. Even at this granularity, relaxation showed great promise, both on our effectiveness experiments and the user study. We believe that finer granularity relaxations will give us better control to cater to user preferences.
- We will explore additional relaxation methods including the one that our crowd users produced unprompted: relaxing base constraints into global constraints.
- We will study the design of more efficient relaxation algorithms that take into account knowledge of data distributions.

Acknowledgements. This paper was presented at the Data4U workshop at PVLDB 2014. This work was partially supported by the National Science Foundation under grants IIS-1421322 and IIS-1420941.

7. REFERENCES

- [1] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Power in unity: Forming teams in large-scale community systems. In *CIKM*, pages 599–608, 2010.
- [2] S. Basu Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu. Constructing and exploring composite items. In *SIGMOD*, pages 843–854, 2010.
- [3] M. Brucato, R. Ramakrishna, A. Abouzied, and A. Meliou. Packagebuilder: from tuples to packages. In *PVLDB*, 2014.
- [4] S. Chaudhuri. Generalization and a framework for query modification. In *ICDE*, pages 138–145, 1990.
- [5] T. Deng, W. Fan, and F. Geerts. On the complexity of package recommendation problems. In *PODS*, pages 261–272, 2012.
- [6] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *Vldb*, pages 199–210, 2006.
- [7] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *KDD*, pages 467–476, 2009.
- [8] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das. On skyline groups. In *CIKM*, pages 2119–2123, 2012.
- [9] A. Parameswaran, P. Venetis, and H. Garcia-Molina. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Transactions on Information Systems*, 29(4):20:1–20:33, Dec. 2011.
- [10] A. G. Parameswaran, H. Garcia-Molina, and J. D. Ullman. Evaluating, combining and generalizing recommendations with prerequisites. In *CIKM*, pages 919–928, 2010.
- [11] K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Transactions on Database Systems*, 36(3):19:1–19:45, Aug. 2011.
- [12] Q. T. Tran, C.-Y. Chan, and G. Wang. Evaluation of set-based queries with aggregation constraints. In *CIKM*, pages 1495–1504, 2011.
- [13] M. Xie, L. V. Lakshmanan, and P. T. Wood. Breaking out of the box of recommendations: From items to packages. In *RecSys*, pages 151–158, 2010.
- [14] X. Zhang and J. Chomicki. Preference queries over sets. In *ICDE*, pages 1019–1030, April 2011.

³Recall that, in the task, preference was given to plans with 4 meals.